07

Comenzando con la herencia

Transcripción

[00:00] Pero ahora, ¿qué pasaría si yo tengo más tipos de trabajadores? Vamos a la presentación. Aquí está mi gerente, pero no necesariamente es el único tipo de trabajador que yo tengo, ya hemos visto que en un banco pueden haber directores, pueden haber contadores, diseñadores, ingenieros, no lo sé, y puede tener tres, seis, o hasta quizás 10 tipos diferentes de trabajador.

[00:38] Ya se imaginarán cómo quedaría el código si yo comienzo a tratar los 10 tipos de trabajador solamente en base a un número. Por ejemplo, ¿quién sabe que el tipo 0 es tipo funcionario? Lo sé yo porque lo acabo de definir y porque lo he comentado aquí. ¿Pero será que para cada tipo preciso comentar mi código y hacer que este if vaya creciendo más y más y más y más?

[01:08] Eso es un antipatrón que ya lo vamos a ver más adelante y lamentablemente ese tipo de prácticas hace que mi código no pueda escalar. Es como que adapto mi negocio al código y no que el código se adapte automáticamente a mi negocio. ¿Por qué? Porque tengo que volver a escribir todo nuevamente, tengo que volver a hacerlo todo nuevamente y pierdo mucho tiempo y mi código es inútil.

[01:34] Mi código, en vez de ayudarme a resolver la vida, me traba y me da más trabajo cada vez. ¿Qué es lo que puedo hacer yo en este caso? Aquí es donde entra el concepto de herencia. Es lo primero que vamos a ver aquí. Y ya hemos visto que por ejemplo vamos a tomar las clases que hemos creado hasta ahora. Vamos a tomar al funcionario y vamos a tomar al gerente.

[02:03] Ya sabemos que los dos tienen nombre, los dos tienen documento y los dos tienen salario. La única diferencia es que cuando los dos van a cobrar su bonificación, a uno le pagan el 10% y al otro le pagan el 100% de su salario. Vamos a identificar el común denominador aquí y son esos tres campos de ahí. Los campos que se repiten. Y vamos a hacer una pequeña analogía. ¿Qué es un funcionario?

[02:40] Un funcionario es simplemente un trabajador, un trabajador cualquiera. ¿Será que el gerente es un funcionario del banco? El gerente es un funcionario del banco, en efecto. El gerente también es un trabajador del banco. El diseñador es un trabajador del banco, el contador es un trabajador del banco. Recuerden muy bien esa analogía ES UN.

[03:10] Eso es lo que les va a ayudar mucho a detectar los casos de herencia. ES UN. Y en Java, por ejemplo, vamos a volver al código, y vamos a volver al gerente. ¿Cómo le digo yo que el gerente es un funcionario? ¿Cómo se lo digo? ¿Cómo le digo public class gerente es un funcionario? Java tiene para esto una palabra reservada que les va a ser de mucha ayuda que es el extend.

[03:56] Public class gerente extiende de funcionario. Esto, entiéndanlo como si dijera extiende de. ¿Por qué? Porque al extender de funcionario yo simplemente puedo borrar todo esto porque funcionario ya lo tiene declarado aquí, funcionario ya tiene todos estos campos, yo no necesito estos campos aquí repetidos, ¿entonces será que lo puedo borrar? Lo borro, perfecto. ¿Será que puedo borrar los getters y setters?

[04:37] Tengo que, porque en este caso ya no tengo más esos atributos ahí y estos métodos no van a compilar. Listo, se fue. Ahora, todo gerente extiende de funcionario. ¿Será que este código sigue compilando? Voy a guardar esto de aquí y TestGerente sigue compilando. Incluso yo podría hacerle aquí un gerente.setNombre y ven aquí como directamente el ID me autocompleta.

[05:17] SetNombre, setSalario, setTipo, setDocumento. Y ustedes dirán: "Pero aquí no hay nada". Espera un momentito. No hay nada, pero gerente extiende de funcionario. Vamos nuevamente a la presentación. Ya que sabemos que el gerente es un funcionario, viéndolo desde el punto de vista de un diagrama de clases podemos primero compararlos.

[05:46] Tenemos funcionario y gerente, y en este caso al gerente yo le puedo hasta añadir más métodos como setClave, autenticar, denegar operación, no lo sé, equis métodos. El gerente puede tener más métodos, pero lo que tienen en común hasta ahora es nombre, documento, salario y getBonificación.

[06:07] Y desde un diagrama de clases el gerente lo que hace es un extend de funcionario donde los datos están declarados en funcionario como nombre, documento, salario, y gerente tiene también los mismos atributos de aquí más algunas cosas propias de un gerente, porque el gerente tiene un usuario, tiene su clave y se puede autenticar.

[06:34] Entonces vamos de regreso al código y vamos a crearles un método setClave. Public void setClave, veamos aquí, y si le estoy creando un setClave tiene que haber un atributo clave, si no, ¿a qué le voy a setear? Entonces le digo aquí un private string clave. Perfecto. Aquí recibo de parámetro una clave y aquí le voy a decir this.clave igual a clave. Esto ya lo conocemos del curso anterior.

[07:27] Y aquí abajo, el método. Vamos a llamarlo iniciar sesión, autenticar creo que no suena muy bien. Iniciar sesión. Y este método como va a retornar si puede o no puede iniciar sesión va a ser un booleano. Perfecto. Y él, como va a iniciar sesión, necesita saber qué clave estás poniendo, entonces como parámetro va a recibir la clave. Listo. Él va a retornar si la clave es igual a, vamos a ponerle una clave muy especial.

[08:19] Esa va a ser la clave. Si la clave es igual a AluraCursosOnLine, entonces él va a retornar un true, acuérdense que esta es una expresión booleana.

Retorna true y da positivo. Si no, da un false y listo, no pasó nada. él no va a poder iniciar sesión. Ahora, yo declaré este método aquí en gerente. Me voy al TestGerente y aquí en set le voy a setear una clave y veo que no tengo el método de resetear clave.

[08:57] Pero Diego, se supone que estamos en gerente, ojo aquí. Tú estás accediendo al funcionario que lo has llamado gerente, y el funcionario no tiene el atributo clave, no tiene el método setClave, a pesar de que el gerente extiende del funcionario, el funcionario no tiene los mismos atributos que el gerente.

[09:22] Entonces vamos a hacer ahí la comparación, ya que el gerente extiende del funcionario, el gerente hereda, acuérdense de esa palabra clave, hereda todas las características de la clase padre, que es el funcionario, pero el funcionario no hereda nada o no obtiene nada del gerente.

[09:48] El gerente obtiene todo del funcionario pero el funcionario no obtiene nada del gerente, el gerente tiene sus propios atributos y sus propios métodos. Por lo tanto, si yo aquí comento esto de aquí y descomento esto de aquí, ¿adivinen qué va a pasar? "Ctrl + espacio" y puedo asignarle una clave. Si yo le doy la clave correcta, vamos aquí a ponerle un System.out.printIn para ver si puedo hacer login o no.

[10:35] Gerente.iniciarSesión, que está aquí abajo y la clave en este caso la voy a copiar nada más, para no escribir tremenda clave otra vez. Guardo todo, le doy play, y true. Conseguí iniciar sesión. Ya hemos visto así entonces una primera aplicación de herencia y un caso práctico muy útil que yo sé que en más de una ocasión nos va a salvar de algún apuro, pero esto recién es la punta del iceberg, tranquilos.

[11:21] Ya vamos a ir poco a poco desenvolviendo más este concepto y dejándolo mucho más claro, así que nos vemos en el siguiente video.